The University of Texas at Austin
Department of Computer Science

Preventing Active Timing Attacks in Low-Latency Anonymous Communication

Joan Feigenbaum, Aaron Johnson, and Paul Syverson

UTCS/TR-10-15
4/26/2010

Preventing Active Timing Attacks in Low-Latency Anonymous Communication

Joan Feigenbaum[1⋆], Aaron Johnson[2⋆⋆], and Paul Syverson[3⋆⋆⋆]

[1] Yale University Joan.Feigenbaum@yale.edu
[2] The University of Texas at Austin ajohnson@cs.utexas.edu
[3] Naval Research Laboratory syverson@itd.nrl.navy.mil

Abstract. Low-latency anonymous communication protocols in general, and the popular onion-routing protocol in particular, are broken against simple timing attacks. While there have been few proposed solutions to this problem when the adversary is active, several padding schemes have been proposed to defend against a passive adversary that just observes timing patterns. Unfortunately active adversaries can break padding schemes by inserting delays and dropping messages.

We present a protocol that provides anonymity against an active adversary by using a black-box padding scheme that is effective against a passive adversary. Our protocol reduces, in some sense, providing anonymous communication against active attacks to providing a padding scheme against passive attacks. It uses timestamping to enforce timing patterns and redundancy to deal with both malicious and benign delays. Because of an asymmetry between sending data to a destination and receiving data from a destination, the protocol uses different techniques in each direction.

Our analytical results show that anonymity can be made arbitrarily good at the cost of some added latency and required bandwidth. We also perform measurements on the Tor network to estimate the real-world performance of our protocol, showing that the added delay is not excessive.

1 Introduction

Anonymous communication protocols are designed primarily to allow *users* to communicate with *destinations* anonymously. They face, however, the challenge of optimizing over several competing criteria: anonymity, latency, and bandwidth. Mix networks such as Mixminion [8] trade off latency for good anonymity guarantees and low message overhead. Protocols such as Dining Cryptographers [6] and Pipenet [7] can provide low latency and good anonymity at the cost of sending a large number of extra messages. Protocols such as onion routing [16]

⋆ Supported in part by NSF grants 0331548 and 0716223 and IARPA grant FA8750-07-0031.
⋆⋆ Supported by NSF grant 0716223.
⋆⋆⋆ Supported by ONR.

and Crowds [31] do not add much latency or message overhead but provide weak resistance to correlation attacks.

High latency and limited bandwidth are unacceptable for many popular Internet applications, and onion routing [16], despite its vulnerability to correlation attacks, has become a successful protocol for anonymous communication on the Internet. To design a useful protocol, we focus on providing better anonymity than onion routing while maintaining acceptable latency and bandwidth.

Low-latency protocols in general have been vulnerable to several attacks based on the timing of events in the system. Typically, the user in these protocols chooses a set of *routers* to mediate between the user and the destination, forwarding data between the two and obscuring their relationship. The essential problem is that timing patterns in these data are conserved between the source and destination. Therefore an adversary only needs to observe the incoming *stream* of data (the consecutive messages exchanged during a communication session), from the user and the outgoing stream of data to the destination to use patterns to link the two. In a *passive* timing attack, an adversary relies on timing patterns that are generated by the user. Because the user creates these patterns, he can prevent this attack by adding dummy packets and delays into the stream to make his traffic look similar to the traffic of other users [34]. However, the adversary can defeat this by performing an *active* attack, in which he inserts timing patterns into the traffic as it passes through routers under his control.

As a result of this sort of active attack, existing low-latency anonymity protocols do not provide anonymity when the adversary controls the routers that the user communicates with directly and the routers that the destination communicates with directly. Suppose the adversary controls a fraction b of the network. In onion routing, users select routers uniformly at random, and the adversary compromises anonymity with probability b^2.

This probability is fixed and cannot be improved by trading off performance elsewhere, and it can be quite insufficient. This is particularly true in a deployed system such as Tor [10], the popular implementation of onion-routing and its associated volunteer network. In Tor, a user sends a message over a sequence of routers he sets up in advance called a *circuit*. For several practical reasons, users in Tor don't choose circuits uniformly at random. Suppose the adversary runs just two routers. If we take into account the way Tor chooses circuits, use the size of the network as of April 2010 [35], and expect the number of daily Tor users observed by McCoy et al. [21], we expect the adversary to compromise 15 different users in one day. If the adversary provides the top two routers by bandwidth, the expected number of compromised users increases to 9464.[1] Thus,

[1] Roughly, circuits are selected in Tor as follows: the first hop is chosen from a set of *guard* routers, the second hop is chosen from the entire network, and the third and final hop is chosen from a set of *exit* routers. As of April 2010, the Tor network consists of around 1500 routers, of which around 250 are guard routers and around 500 are exit routers. Suppose that the adversary runs one guard router and one exit router. McCoy et al. observed 7571 unique clients while running a guard router for one day. The expected number of these that would lose anonymity is $7571/500 =$

the system provides poor anonymity against a wide variety of realistic opponents, such as governments, ISPs, and criminals willing to purchase the use of botnets.

We consider the very weak anonymity provided by low-latency protocols against an active adversary to be a fundamental and critical challenge in anonymous communication. In this paper, we present a low-latency protocol that provides arbitrarily good anonymity against an adversary that can observe and create timing patterns. The protocol makes black-box use of a padding scheme to prevent passive timing attacks. Several padding schemes that defeat passive timing attacks have been proposed [34, 32, 37], and furthermore we believe that there is still potential for substantial improvement. The protocol provides two-way stream communication.

A two-way protocol requires different defenses, depending on the direction of communication, because of an asymmetry in the communication between the user and destination. The user can talk directly to many routers and will add padding correctly. We will require that the destination communicate with just one router, and that router can't be trusted to pad the stream correctly. As a result, our protocol uses a somewhat different scheme for traffic on the way *from* the user as on the way *to* the user. The essential features of our solution are

1. We add timestamps to some packets indicating their intended send time.
2. Packets from the user are sent in several copies over a *layered mesh* topology . This balances providing redundancy against malicious delays against limiting view of the stream to a small number of routers.
3. Packets from the destination to the user are sent over a path that performs in-stream padding.

For simplicity, we describe forming the layered mesh as a *cascade*: a fixed arrangement of routers that all users use to send data. The biggest drawback to using cascades is that the resource constraints of the cascade routers obviously limit the number of feasible users and therefore limit anonymity. Another drawback is that cascades make long-term intersection attacks easier because only two known endpoints need to be watched. Giving users freedom to choose the meshes, analogous to *free routes*, is an important future extension to our scheme.

We evaluate the anonymity provided by our protocol in a network model that incorporates timing and an active adversary. The theoretical results suggest that the approach has good asymptotic efficiency and that a promising next step is to optimize within the framework of the scheme we describe. Moreover, because we are concerned with eventual practicality, we do measure a component of the

15.142. Moreover, Tor weights by bandwidth, and so suppose that the adversarial routers are the top two by bandwidth. McCoy et al. observed that the top 2% of routers transported about 50% of the traffic. Then, very roughly, the probability of choosing the adversary in a guard set would increase from 1/250 to .5/(250*.02), and so the expected number of users observed would be 25*7571=189275. By similar rough approximation, for every circuit, the adversary's exit router would be selected with probability .5/(500*.02)=.05, and so the expected number of deanonymized users would be 189275*.05=9463.75.

system over which we have no control and which could have made our protocol unusable, delay variations in the host network. Specifically, we measured the likely latency costs of running our protocol on the existing Tor [10] network. This provides a strenuous, real-world scenario to evaluate our protocol's performance.

Our results are therefore both theoretical and experimental:

1. We show that the user can be made anonymous with arbitrarily high probability as long as b is less than $1/2$. The user is anonymous within the set of users with identical traffic patterns as produced by the input padding scheme.
2. We prove that our mesh topology in the forward direction is optimal for anonymity in a limit sense.
3. The latency of our protocol is proportional to the length of the mesh and return path. We show that the probability of compromise decreases to zero polynomially in that length. This compares well with onion routing, which adds latency proportional to its path length.
4. The bandwidth used is $2w + (l-1)w^2 + 1$, where l is the mesh/path length, and $w = \Theta(\log l)$. This compares well asymptotically to the $l+2$ copies that are sent in an onion-routing path of the same total length.
5. For most of our measurements, we observe that added packet delay would need to be less than a factor of two to achieve satisfactory reliability.

The results suggest that our approach indeed has the potential to mitigate active timing attacks. An extended abstract of this Technical Report appears in the Proceedings of the 2010 Privacy-Enhancing Technologies Symposium.

2 Related work

Timing attacks are a major challenge in low-latency anonymous communication [20]. They have been observed in some of the earliest low-latency systems [1], including initial versions of onion routing [16]. These attacks are also closely related to traffic analysis in mix networks [30].

In a passive timing attack, the adversary observes timing patterns in a network flow, and then correlates them with patterns in other traffic that it observes. If the adversary is able to observe both the user and the destination, he can thereby link the two. The ability of the adversary to perform this correlation has been experimentally demonstrated several times [39, 20, 29, 27, 2].

A solution to passive timing attacks is to get rid of identifying patterns in the traffic by padding and delaying it. The drawbacks to such an approach are added latency and bandwidth overhead. Our protocol relies on the existence of some acceptable and effective padding scheme. Constant-rate padding, in which traffic is sent at a constant rate by filling in the gaps with dummy packets, is probably the most obvious such scheme. It has appeared multiple times in the literature [34, 15, 20]. Levine et al. [20] propose a "defensive dropping" mechanism which adds dummy packets at the start of the circuit and drops them at various routers before the end. This reduces the correlation between any patterns

in the incoming streams and patterns in the outgoing streams. Shmatikov and Wang [32] propose a variable-rate padding scheme. In their scheme, packets from the user are forwarded with little delay, and dummy packets are added by the intermediate routers according to a probability distribution on the packet inter-arrival times. Wang et al. [37] describe a link-padding scheme for low-latency systems, but their system is designed for a situation in which the adversary is not active and the destination participates in the protocol. This situation does not reflect the kind of Internet communications that have proven useful and that we target.

All of these schemes are vulnerable to an adversary that actively delays packets from the user. Yu et al. [38] show that this can be done in a way that makes it difficult for the user or the system to detect that the attack is occurring. One approach to this problem is to change the timing patterns within the network by adding, dropping, or delaying packets ([32, 20]). However, dropping or delaying packets can't hide very long delays without adding unacceptable latency or bandwidth overhead. Adding dummy packets can, in our case, be detected by the final router, and therefore do not help. A final router can detect them because, in our case, the destination does not participate in the protocol; there must be one last router in the system that provides the point of contact to the destination. Moreover, Wang et al. [36] experimentally show that many such schemes for internal traffic shaping are still vulnerable to an active timing attack.

Simply delaying packets that pass directly through adversarial routers isn't the only active timing attack that has been demonstrated. Murdoch and Danezis [26] show that in onion routing the adversary can actively add delay patterns to the data by sending bursts of traffic through a router. This can be used to determine the routers on a given circuit. A queuing scheme has been suggested by McLachlan and Hopper [22] to mitigate this problem. Fu et al. [14] describe how the presence of a flow on a router can also be determined by ping times to the router, and suggests the simple countermeasure of treating ping packets equally. Borisov et al. [4] look at the case that the adversary doesn't just delay, but drops packets in a denial-of-service (DoS) attack aimed at forcing users to move to circuits that the adversary can deanonymize. This is an easy behavior to observe, although proving the culprit may be difficult, and the authors suggest that reputation mechanisms can be used to limit such attacks. Such an attack was also discussed by Dingledine et al. [11]. We do not address congestion and DoS attacks in this paper, and they are outside of our model.

A related timing attack by Hopper et al. [17] uses congestion to exploit different network latencies between hosts. They show that the latencies from multiple hosts to a user can be very identifying. The user in our protocol communicates with several routers as a first hop, and in light of this attack we took care not to allow the adversary to infer these latencies.

One key feature of our protocol is the use of a layered mesh to provide redundancy. The use of redundancy for several purposes has been also explored in previous protocols. Syverson [33] suggests using router "twins," pairs of routers that share the same key, to provide a backup in case a router fails. Iwanik et

al. [18] also use redundancy to avoid failures in their K-Onion scheme, in which onions are encoded in a way that provides more than one option for the next hop. Redundancy to support failure is not our goal, and such schemes are in some ways complementary to our own. However, the redundancy in our protocol does protect against honest node failures as well as malicious ones. Iwanik et al. also describe the Hydra-Onion scheme, in which onions get forwarded to two routers at every hop instead of one. The purpose of this scheme is to prevent an active attack in which the adversary kills messages from a known user and observes which destination stops receiving traffic. This situation is quite similar to our own. However, they allow the destination to participate in the protocol, and the scheme is designed for an adversary that controls links instead of routers. Finally, Nambiar and Wright [28] use redundancy in the host lookup of Salsa to protect against route capture. Interestingly, an analysis by Mittal and Borisov [24] of this technique uncovers the tradeoff between preventing active and passive attacks that we face as well.

Another critical feature of our protocol is the use of explicit timing to coordinate the users and routers. This is similar to the timing instructions of Stop-and-Go mixes [19]. Such mixes are given a time window within which the packets must arrive, and they delay forwarding by an exponentially-distributed amount of time. Although the techniques are similar, this scheme is designed for mix networks and not stream communication, and this scheme does give the adversary some slack time within which a timing signature could possibly be placed. Moreover, the lack of any redundancy means that any slowdowns within the network, even of benign origin, can quite easily kill a connection.

The timing approach we take is also similar to the use of synchronicity in mix networks by Dingledine et al. [11]. They describe synchronous batching on free routes and show that it typically provides better anonymity than cascades. In their case the synchronicity is provided by long delays in the mix network, while in our case synchronicity is provided by the padding schemes. They are able to disregard active attacks in which messages are dropped or delayed, because each mix is able to wait for enough time that any missing packets must be maliciously withheld. They rely on reputation systems [12] to eventually find and punish such action. With low-latency communication, on the other hand, we cannot assume that such delays will not happen. Moreover, our solution does not rely on the use of an external reputation mechanism. This is an advantage, because reputation mechanisms in general only detect large-scale misbehavior and must tolerate smaller-scale targeting of users.

3 Model

We will express and analyze our anonymity protocol in a model of network and adversary behavior. A particular advantage of this approach is the ability to make convincing guarantees of security when we cannot predict the tactics that an adversary will use.

3.1 Network

Let the network consist of a set of onion routers R, a user population U, and a set of destinations D. The network is completely connected, in that every host can send a message directly to every other host. Each event in the network occurs at some global time. We assume that each user and router has a local clock that accurately measures time, and that these clocks have run some sort of synchronization protocol [23]. Let δ_{sync} be the largest difference between two synchronized clocks in the network.

There is some probabilistic network delay $d_{net}(r, s)$ between every pair (r, s) of routers. This models the unpredictable delay between routers due to factors such as route congestion and route instability. There is also some probabilistic processing delay $d_{proc}(r)$ at every router r. This reflects changes in delay at a router due to local resource contention among anonymous messages and among multiple processes. The delay of a message is the sum of delays drawn independently from these two distributions. We also let the delay of one message be independent of the delays of the other messages. We assume the distributions of both sources of delay is known to the system. In practice, this may be achievable via network measurements.

We assume that all hosts (in particular, all destinations) respond to a simple connection protocol. One host h_1 begins the connection by sending to another host h_2 the pair $< n, M >$, where $n \in N^+$ is a number and M is a message. Any responses M' from h_2 are sent back as $< n, M' >$. Once established, connections in the anonymity protocol cannot be closed by anyone to prevent distinctions of one from another by open and close times. All connections thus stay open for a fixed amount of time and then close automatically. Application connections running over these can of course be closed by the ultimate source and destination; although this will not close the anonymity circuit, and padding messages will continue to be sent until connection timeout.

3.2 Users

User communication drives the operation of the anonymity network. We view the communication of a user as a sequence of connections. Each connection is to one destination, and it includes messages to and from the destination. 'User' refers to both human users and software running on their behalf.

3.3 Adversary

The adversary controls some subset $A \subseteq R$ of the routers, where $b = |A|/|R|$. It seems plausible that an adversary can run routers that are at least as fast as the other routers on the network, and that it may dedicate them to running the protocol. Therefore, in contrast to non-adversarial routers, we pessimistically assume that the adversary has complete control over the processing delay $d_{proc}(a)$ of routers $a \in A$. If needed, we reflect compromise of links between routers by the compromise of an adjacent router.

3.4 Padding scheme

The padding scheme $\mathcal{P}$ is a black box that initially takes as input a connection start time and outputs the timing of the return traffic. Then every time step it takes the presence of data from the user and returns whether or not a packet should be sent. Note that this requires the scheme to determine in advance the length of the connection. Let S_u be the set of users that start connections at the same time as user u and have the same traffic pattern. Our proposed protocol relies on the effectiveness of the padding scheme. At best, it makes u indistinguishable within the set S_u supplied by $\mathcal{P}$.

Some of the padding schemes previously proposed in the literature can provide the black box $\mathcal{P}$. For example, to use basic constant-rate padding, in which packets get sent at constant rate in both directions, we simply need to choose a fixed length for the connection when it starts. This approach typically causes high added latency and/or message overhead, however. As another example, the padding scheme of Shmatikov and Wang [32] could be used by fixing the connection length and return scheme. In this padding scheme, inter-packet delays are sampled from a distribution, which is adjusted if a packet arrives early. Dummy packets are sent after the sampled delay, and real packets from the user are sent immediately. In the direction from the user, this scheme could be used directly. In the return direction, we could just skip shifting the distribution for early packets. Then we would send each return router the same sequence of random bits to use in sampling the distribution. Alternatively, we could relax our requirements for the return scheme and allow it to be updated periodically. The user could then use the forward mesh to update the distribution of packet arrival times.

It is not hard to conceive of novel padding schemes that might satisfy these requirements, although getting a good mix of anonymity and performance certainly does not seem easy.

4 Problem

The problem in this model is to design an anonymity protocol that supports the low-latency, two-way, stream communication that has made Tor [10] popular. We understand stream communication to mean that users communicate in sessions of arbitrary length and volume, which makes embedding timing signatures possible. In order to allow communication with hosts that are ignorant of the protocol, we require that only one host communicates with the final destination, and that the communication is only the original messages generated by the user. We evaluate our protocol by three criteria: anonymity, latency, and the amount of data transferred. We evaluate our protocol according to its *relationship anonymity*, that is, the extent to which it prevents an adversary from determining which user-destination pairs are communicating. For latency, we consider the amount of time it takes for a message to reach the destination from a user. For the amount of data transferred, we consider the total amount of data that has to be transferred during a single user connection.

5 A Time-stamping Solution

The padding scheme gives us sets of users that have traffic streams with identical timing patterns. However, the model we have described gives the adversary the ability to modify these patterns as the traffic travels through its routers towards the destination. To prevent this, we try to enforce the desired timing pattern on packets sent by including the times that the routers should forward them. Any honest node that receives the packet will obey the instructions, removing any delays inserted by the adversary. For traffic sent from the user to the destination we can trust the user to correctly encode the padding-scheme times. Traffic sent from the destination to the user must, by our requirements, pass initially through a single router. Because it may be compromised, we cannot trust it to use a proper padding scheme. However, this traffic is destined for an anonymity-protocol participant, the user; therefore, unlike traffic from the user, we can destroy inserted timing patterns by re-padding it all the way to the user. Observe that re-padding does not work for traffic from the user, because the final router sees which packets are real and which are padding.

5.1 From the user

First, consider what we could do if propagation and processing delays were deterministic. The user could send through a path in the network a layered data structure called an *onion* which, for each packet, includes in the ith layer the time that the onion should arrive at the ith router. Then each router on the path could unwrap the onion to make sure that the initial timing sequence was being preserved and, if so, forward the onion.

Unfortunately, in real networks, delays are somewhat unpredictable. For example, an onion might be delayed by congestion in the underlying network. However, if the distribution of delays is known, we know how long we need to wait at a router for onions to arrive with any fixed probability. We will set that probability to balance decreasing added latency with decreasing the chance of a successful timing attack. Then we add in this buffer time to the send time.

Another problem is that the adversary could drop onions entirely in a pattern the propagates down the path. Our approach to this problem is to send multiple copies of an onion down redundant, intersecting paths. A router on a path needs only one copy of the onion to arrive in time from any incoming path in order to forward it by its send time.

This approach has limits, because each redundant router adds another chance for the adversary to observe an entire path from source to destination. For example, suppose that we simply send onions over k paths of length l that intersect at a final router, where every router is chosen uniformly at random. Let b be the fraction of routers that are controlled by the adversary. The probability that at least one path is entirely composed of compromised routers is $b(1-(1-b^{l-1})^k)$. This quickly goes to b as k increases. We use a layered-mesh topology to balance the ability of the adversary to passively observe a path with his ability to actively perform a timing attack.

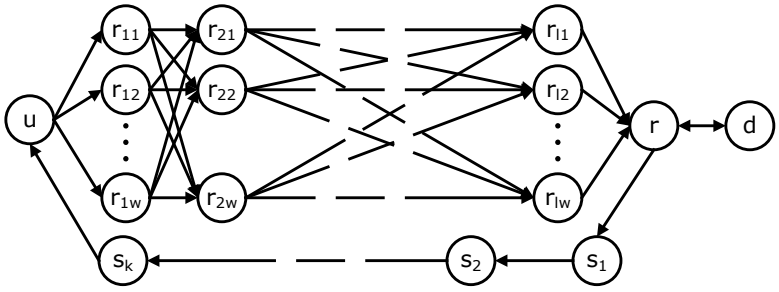


Fig. 1. Layered-mesh topology

Topology The layered-mesh topology we propose is pictured in Figure 1. For some length l and width w, user u sends a copy of each onion to the w members r_{1j} of the first layer. Then, in layer i, each router r_{ij} sends one copy of every onion it receives to each router $r_{(i+1)k}$ of the next layer. Finally, the routers r_{lj} in the last layer send a copy of each onion received to a single router r, which finishes decrypting them and sends the data on to the destination d. We call this structure the *layered mesh*.

Timestamps As described, the user sets timestamps to instruct routers to maintain a specific timing pattern. A user may have different latencies to the different routers in the first layer. If the time that one of these routers is instructed to forward the packet only depended on the network and processing delays of that router, the first-layer routers could send copies of the same packet at different times. This would provide information to the next layer about the identity of the user. Similarly, the adversary could use different times between layers to link other layers in the mesh. Therefore, we set the send time for each layer to be the send time of the previous layer plus the longest delay at our chosen reliability level p. We must also add some extra delay to allow for clock skews.

Let $d^*(r, s)$ be the amount of delay we need to ensure that a packet from r is processed at s with success probability p:

$$p = Pr[d_{net}(r, s) + d_{proc}(s) \leq d^*(r, s)].$$

At time t, let user u be instructed by $\mathcal{P}$ to send a message. The user chooses the same send time for all routers in the same layer. The send time for routers r_{1j} in the first layer is

$$t_1 = t + \max_j d^*(u, r_{1j}) + \delta_{sync}.$$

The send time for routers r_{ij} in the ith layer is

$$t_i = t_{i-1} + \max_{j,k} d^*(r_{(i-1)j}, r_{i,k}) + \delta_{sync}.$$

If a router receives its first copy of an onion after the send time has passed, it immediately forwards the onion to the routers in the next layer. At worst,

the delay is the result of attempts by the adversary to delay certain packets. Sending the packet later or not at all in that case would only make it easier for the adversary to observe its lateness later in the mesh. Forwarding it immediately might even allow the onion to eventually get back on schedule. At best, the delay is just a result of network delays and forwarding has no effect on anonymity.

Onions Let M be the message to be sent. We will encrypt the message with a public key shared by all members of a layer. Given that the layers are set up in advance and known to all, such a key can be generated by a trusted third party or by electing a leader to do it. Let $\{M\}_{r_i}$ denote the encryption of M with the public key of layer i. Let $n_{r_i}, n_{s_1} \in \mathbb{N}$ be random numbers and k_r be a private key. Then the onion that u sends to the routers in layer 1 is

$$\{n_{r_1}, t_1, \{n_{r_2}, t_2, \cdots \{n_r, d, n_{s_1}, k_r, M\}_r \cdots\}_{r_2}\}_{r_1}$$

For each layer i, a user generates the random number $n_{r_i} \in \mathbb{N}$ as an onion identifier. The routers keep track of the onion identifiers they have seen. When they receive an onion, they decrypt it and examine the identifier. Routers only forward an onion if its identifier n_i has not been seen before. n_{s_1} is the identifier that r should use with s_1 when sending back any reply, and k_r is a private key that will let r encrypt the return message for u.

The onion encoding and forwarding scheme should hide routing information, prevent forgery, prohibit replay attacks, and hide message content. For clarity of presentation, we have described a simple scheme that achieves this. We observe, however, that several improvements to the protocol could be made. For example, the protocol could do a more explicit stream open and close to reduce the lists of identifiers that routers have to maintain. Also, symmetric keys could be exchanged to speed up onion processing. Another improvement that we could incorporate is forward secrecy. Numerous cryptographic details must be carefully set out (*e.g.* as in [5]) for our protocol to have a cryptographically secure and efficient implementation. These are not the focus of this paper.

5.2 To the user

Traffic returning to the user from the destination must first pass through the one router that is selected to communicate directly with the destination. This router may be compromised, and it may try to insert timing patterns in the return traffic. We manage this by giving the intermediate routers the pattern of the return traffic. They enforce it by fitting the return onions into the pattern, adding dummy packets when necessary. We note again that this doesn't work for the traffic from the user because any added delays translate into delays in the underlying data, and this can be viewed by the final router. We choose a simple path of length k for the return traffic (Figure 1), because there is no anonymity advantage to adding redundancy here. We call this structure the *return path*.

To communicate the desired traffic pattern to the return path, we take advantage of the one-way communication protocol already developed. The user

takes the return traffic pattern that is given by the padding scheme $\mathcal{P}$ and sends it via the layered mesh to every router in the return path. At the same time, the user generates the random numbers $n_{s_i} \in \mathbb{N}, 1 \leq i < k$, and sends two random numbers $n_{s_i}, n_{s_{i+1}}$ and a key k_{s_i} to each router s_i. The numbers will be the incoming and outgoing identifiers for the onion. The user also sends n_k and u to s_k. Let M be the message to be sent back from d to u. The return onion sent to s_i is

$$O_i =< n_{s_i}, \{\cdots\{\{M\}_{k_r}\}_{k_{s_1}}\cdots\}_{k_{s_{i-1}}} > .$$

After r receives M from d, it will take n_{s_1} and k_r out of the inbound onion from the user and send O_1 to s_1. When s_i receives O_i it looks for a matching n_{s_i} in the pairs of number it has received and then forms O_{i+1} to send when the padding scheme instructs it to.

5.3 Choosing the routes

We let the layered mesh and return path be chosen in advance and shared among all users. This is analogous to cascades in mix networks [3]. To balance the load across the network, multiple such mesh-path pairs can be chosen. Users then randomly select a pair. Our analysis applies to all users choosing the same pair. A trusted third party can be used to select the pair; this is a reasonable option for Tor. Otherwise, protocols for the network to select cascades are described in [12]. The routers in the topology are picked uniformly and independently at random.

A disadvantage of cascades is that number of users that can be confused with one another, *i.e.*, the size of the anonymity set, is at most the number of users that can simultaneously be handled by a router [11]. Allowing users to choose the cascades with some freedom should allow anonymity sets to grow with the size of the network, similar to free routes in onion routing, but we leave this to future work.

Though outside of the threat model we consider in this paper, one can consider an attacker that attempts to compromise specific nodes over an extended period in an attempt to increase the chances of owning an entire layer and an entire path from a client to that layer. Against such an attacker one would want to have a dynamic network structure in which the positions that layers can be in change periodically as does the composition of each layer group. These may or may not change at the same rate. It is beyond the scope of this paper to do more than note the existence of such responses to a long-term strategically roaming attacker.

6 Analysis

The purpose of our protocol is to provide better anonymity than onion routing at reasonable cost in latency and bandwidth. A major drawback to onion routing is that the probability of compromise is b^2 and cannot be improved, *e.g.* by choosing

a longer path. We will show how in fact our scheme can provide arbitrarily good probability by increasing the length and width of the layered mesh.

First, the design of the protocol onions essentially limits the adversary to traffic analysis. For traffic from the user, the use of encryption and unique identifiers forces onions to be passed through the required layers and limits them to being forwarded by an honest router at most once. It also hides the source and destination from all but the first and last routers, and it makes messages leaving one layer unlinkable to messages leaving another layer. For traffic to the user, the source and destination are not included in the packets, and encryption prevents messages leaving one router from being linked with messages leaving another router.

For the adversary to break a user's anonymity, then, he will need to either observe traffic on an entire path between source to destination or link traffic at different steps on that path. The latter depends on his ability to introduce delays in the packet stream. To evaluate this possibility, we will make the simplifying assumption that the network and processing delay between two routers r, s never falls above the time allowed for it $d^*(r, s)$. In our model, such failures can happen with probability $1 - p$, where p can be set arbitrarily close to 1. Certainly, if the adversary can deanonymize a user even under this assumption, he can do so with possible link failures. When such failures do occur, they open up the chance for an adversary to successfully delay packets and insert a timing pattern. However, late packets are typically irrelevant because the same packet will have been forwarded by another router in the previous layer. Also, late packets that are the first of their type to arrive are immediately forwarded, thus benign delays are somewhat self-correcting, and we estimate that they do not open up a large opportunity for an active timing attack. However, if we wish to make a conservative estimation, we can expect that for any given packet a fraction $1 - p$ of the packet copies will fail to arrive in time. We can estimate the combined probability of malicious or benign delay or dropping of packets by $(1 - p) + b$. Of course, p changes with every packet, but for connections with few packets from the user, such as web connections, this approximation may still yield good anonymity guarantees.

Assuming no link failures, then, the anonymity of a user only depends on which routers the adversary controls. Because all users on the same cascade use the same routers, the adversary can either deanonymize all users in the anonymity set S_u, or he can not deanonymize any of them. Because the routers in the cascade are selected randomly, there is some probability that the adversary can deanonymize the users. Let $\mathcal{C}$ be the event that the adversary can compromise anonymity. We can quantify the probability of $\mathcal{C}$.

Theorem 1

$$Pr[\mathcal{C}] = b^{k+1} + b(1 - b^k)\left[b^w \frac{1 - (1 - (1 - b)^w - b^w)^l}{b^w + (1 - b)^w} + (1 - (1 - b)^w - b^w)^l\right]$$

Proof. To compromise anonymity, the adversary must control the router r that communicates with the destination. Therefore assume that $r \in A$. This happens with probability b.

Then he can link the destination via the return path only when he controls every router on it. This happens with probability b^{k+1}.

The adversary can deanonymize the user via the mesh in two ways. First, the adversary can control an entire path from the first layer in the mesh to r, but not any entire layer. This happens with probability $(1-(1-b)^w-b^w)^l$. Second, if the adversary controls a path to a completely controlled layer i, he can then use it to selectively delay packets from one user. He can identify the user to target because he controls a path from that user to layer i. This happens with probability $b^w(1-(1-b)^w-b^w)^{i-1}$. If neither of these is the case, then it must be that there is an honest path in the mesh leading to a completely honest layer. This layer receives all packets from all users via the honest path, and beyond it the adversary cannot distinguish between users. Therefore deanonymization is not possible.

Adding all of the probabilities, we get

$$Pr[\mathcal{C}] = b^{k+1} + b(1-b^k) \cdot \left[b^w \sum_{i=0}^{l-1} (1-(1-b)^w-b^w)^i + (1-(1-b)^w-b^w)^l \right]$$

Simplifying the partial geometric sum gives the theorem. □

Theorem 1 shows that, when less than half of the routers are compromised, we can make the probability that a user is anonymous arbitrarily high by setting w, l, and k large enough. (Recall that all proofs are in our technical report [13].)

Corollary 2

$$\lim_{w,l,k\to\infty} Pr[\mathcal{C}] = \begin{cases} 0 & b < 1/2 \\ 1/4 & b = 1/2 \\ b & b > 1/2 \end{cases}$$

Proof. As $k \to \infty$, the first term in the expression of Thm. 1 goes to zero. As $k, l \to \infty$, the second term goes to $b^{w+1}/(b^w + (1-b)^w)$. When $b < 1/2$, this goes to zero with w, when $b = 1/2$, this is $1/4$, and when $b > 1/2$, this goes to b. □

Corollary 2 shows that anonymity can be made arbitrarily good when $b < 1/2$, but that it is worse than onion routing when $b \geq 1/2$. Therefore assume from now on that $b < 1/2$. We would like to determine how big the layered mesh and return path grow as we increase our desired level of anonymity. First, we will consider how wide the mesh must be for a given depth to achieve optimal anonymity. This affects the number of messages that need to be sent. Also, it will allow us to evaluate anonymity as a function of the lengths k and l, quantities which we would like to keep small to provide good latency. Luckily, it turns out that the optimal width w^* grows slowly as a function of l.

Theorem 3 $w^* = O(\log(l))$

Proof. Let $\mathcal{A} = \mathcal{C}^c$ be the event that users are anonymous. From Thm. 1, we can determine that

$$Pr[\mathcal{A}] = 1 - b + b(1-b)^w \frac{1 - (1 - b^w - (1-b)^w)^l}{b^w + (1-b)^w}. \tag{1}$$

Let x be the third term in Equation 1. It is the only term that varies with w. Because $b < 1/2$, we can replace b with $1-b$ at point in x to obtain a simpler upper bound:

$$x \leq b(1 - (1 - (1-b)^w)^l). \tag{2}$$

Similarly, we can replace $1-b$ with b to obtain a lower bound:

$$x \geq b(1 - (1 - (1-b)^w)^l)/2. \tag{3}$$

Equation 2 is decreasing in w. Equation 3 is increasing for small w, achieves a maximum at $w = \log(1/2)/\log p$, and is decreasing for larger w. At maximum, Equation 3 has value $b/2$. Therefore $Pr[A]$ must be maximized at a smaller w than that at which the upper bound in Equation 2 reaches $b/2$. We can further approximate to find this w:

$$x \leq b(1 - (1 - (1-b)^w)^l) \tag{4}$$
$$\leq bl(1-b)^w \tag{5}$$

Expression 5 is $b/2$ when w is $\log(2l/b)/\log(1/(1-b))$. □

Thm. 3 shows that the total number of messages sent for every message from the user is $O(2\log(l) + (l-1)\log(l)^2 + 1)$. This compares well asymptotically to the $l+2$ copies that are sent in an onion-routing path of the same total length.

Network users are particularly sensitive to latency, and each additional step in the layered mesh and return path represents a potentially global hop on our network. To keep the lengths l and k of the mesh and return path small, then, we would like for $Pr[\mathcal{C}]$ to converge quickly to its limit. As suggested by Thm. 3, let $w = \log l$, and consider the convergence of $Pr[\mathcal{C}]$. It is clear that the first term shrinks exponentially with k. The convergence time of the second term is polynomial, although it does improve as b gets smaller.

Theorem 4 *Let $c_1 = \log(b)$ and $c_2 = \log(1-b)$. Then*

$$Pr[\mathcal{C}] = \Theta(l^{c_1 - c_2}).$$

Proof. Let us fix k and only consider how $Pr[\mathcal{C}]$ changes with increasing l. Let x be the part of the expression in Thm. 1 that varies with l and substitute in $w = \log l$:

$$x = bl^{c_1} \frac{1 - (1 - l^{c_1} - l^{c_2})^l}{l^{c_1} + l^{c_2}} + b(1 - l^{c-1} - l^{c_2})^l.$$

The first term in x is less than l^{-c_2}. We can give an upper bound for the second term by using the inequality $1 - y \leq e^{-y}$:

$$b(1 - l^{c-1} - l^{c_2})^l \leq b(1 - l^{c_2})^l \leq be^{-l^{1+c_2}}$$

$1 + c_2 > 0$ because $b < 1/2$. Therefore $x \leq b(l^{c_1 - c_2}) + e^{-l^{1+c_2}} = O(l^{c_1 - c_2})$.

We can give a lower bound for the first term in x by using the fact that $1 - y \geq e^{-2y}$ for $y \leq 1/2$:

$$bl^{c_1} \frac{1 - (1 - l^{c_1} - l^{c_2})^l}{l^{c_1} + l^{c_2}} \geq bl^{c_1} \frac{1 - e^{-2(l^{c_1} + l^{c_2})l}}{2l^{c_2}} \geq \frac{b}{2}(1 - e^{-2l^{1+c_2}})l^{c_1 - c_2}$$

Therefore $x = \Omega(l^{c_1 - c_2})$. □

Table 1 compares the performance of our mesh topology to that of onion routing using some reasonable parameter values. In it, we let both the mesh length and the onion-routing-path length be l, we let the length of the return path from the mesh equal the mesh length (*i.e.* $k = l$), and the width w of the mesh is set to optimize anonymity. We use small values of l to make the number of hops close to the three hops that have proven to be usable in the current Tor system. The numbers show clear decreases in the probability of compromise when using the mesh, especially with larger values of l. We can see that larger compromised fractions b will require somewhat longer paths for significantly improved anonymity. The total number of messages sent in each scheme for every message-response pair between the user and destination is also given.

			Mesh		**Onion Routing**	
b	l	w	$Pr[\mathcal{C}]$	Msgs.	$Pr[\mathcal{C}]$	Msgs.
.05	3	3	.0002	29	.0025	8
.05	4	3	.00003	39	.0025	10
.1	4	3	.0007	39	.01	10
.25	4	2	.0303	22	.0625	10

Table 1. Mesh routing vs. Onion routing

Our analysis shows how our scheme can provide arbitrarily good probability for $b < 1/2$. Is it possible to improve this to include values of b greater than $1/2$? First, we observe that some other plausible topologies do not perform as well. For example, suppose the user sends onions to k routers, each of which forwards it to the same next router, and then from then on there is a path to the destination. The probability that anonymity is compromised in this situation is $b^2(1 - (1 - b)^k + b^{k-1}(1 - b))$ (see Appendix A.1). As k grows, the anonymity goes to b^2, the probability that the second-layer router and final router are both compromised. As another example, consider using a binary tree, where the user sends to the leaves of the tree, and each node forwards to its parent at most one copy of the onions it receives. As the depth of the tree increases, the probability that anonymity is compromised goes to zero when $b \leq 1/4$, $b(4b - 1)$ when $1/4 \leq b \leq 1/2$, and b when $b \geq 1/2$ (see Appendix A.2).

The following theorem shows that the layered mesh is optimal in the limit among all topologies.

Theorem 5 *Let $c(b)$ be the probability of anonymity compromise in some forwarding topology when the fraction of adversarial routers is b. Then, if $b < 1/2$, $c(b) < \epsilon$ implies that $c(1 - b) > 1 - b - \frac{1-b}{b}\epsilon$.*

Proof. It is easy to show that we can assume the following: the topology is acyclic, there are two users, u and v, and the adversary's strategy is to forward only v's packet when he is sure it belongs to v and otherwise to block all packets.

Assume that the router communicating with the destination is adversarial. Then, for a given topology, let α be a Boolean function on the remaining router positions that indicates if that position is controlled by the adversary or not. If anonymity is not compromised under α, there must be a set of honest routers C such that C forms a cut between the users and destination and every router in C sends packets from u and v. To see this consider the induced subgraph of routers that send the packet from v but not from u. It cannot connect u and the last router or anonymity would be compromised. Therefore, there must be a set of routers that are outgoing neighbors to this subgraph. These routers and the honest first-hop routers constitute the desired cut.

Every router in C must have a path of honest routers from the user. Otherwise there would be a set of adversarial routers that cut a router in C from the users, and no packets from the users would reach that router. Now consider the assignment $\overline{\alpha} = 1 - \alpha$. If α provides anonymity, then in $\overline{\alpha}$ there must be a cut consisting of adversarial routers such that each router has a path of adversarial routers from the users. Thus anonymity is compromised in $\overline{\alpha}$.

Let $c'(b)$ be the probability of anonymity compromise given that the final router is compromised. The probability of assignment α with a fraction b of compromised routers is equal to the probability of assignment $\overline{\alpha}$ with a fraction $1 - b$ of compromised routers. Because the complement of an assignment providing anonymity is an assignment the compromises anonymity, we have

$$1 - c'(b) \leq c'(1 - b). \tag{6}$$

$c(b) = bc'(b)$, and, therefore, if $c(b) < \epsilon$, then $c'(b) < \epsilon/b$. Combining this with the previous inequality Inequality 6, we get that $c'(1 - b) > 1 - \epsilon/b$, which implies that $c(1 - b) > 1 - b - \frac{1-b}{b}\epsilon$. □

Theorem 5 implies that if the probability of compromise for a topology goes to zero for $b \in [0, \beta]$, then it must go to b for $b \in [1 - \beta, 1]$. This is achieved by the layered-mesh topology for the largest range of b possible, $[0, 1/2)$.

7 Latency Measurements

Our protocol requires routers to hold messages until a specified send time. Latency between routers varies, and therefore this send time must be set high enough to guarantee that with sufficiently high probability that it occurs after the packet actually arrives. The amount of time that packets spend being delayed before the send time depends on the variability of latency. Because network performance is critical to realizing effective anonymous communication [9], we wish to evaluate the latency in our protocol.

In order to do so, we have measured latency in the Tor [10] network. Performance data in the Tor network gives us reasonable estimates for the performance of our protocol for several reasons. First, the essential router operation in both protocols is decrypting and forwarding packets. Second, the network is globally distributed and therefore includes a wide variety of network and host conditions.

Third, using volunteers to provide the routers is a major reason for the success of Tor; we might reasonably expect future networks to also be run by volunteers and therefore have similar computational and communication resources.

7.1 Methodology

During each experiment, we made three measurements on Tor routers:

1. We measured the *round-trip time* (RTT). To do so, we opened five TCP connections in a row and measured the time between the TCP SYN request and the SYNACK response.
2. We measured the *connection delay*, To do so, we created a circuit from a host on the Yale network to the router and opened a TCP stream over that circuit from the router back to the Yale host. We measured the time between sending the stream-open request to the router and receiving the TCP connection from the router.
3. We measured the *packet delay*. To do so, we sent five 400-byte segments of data up the established TCP stream. The segments were sent every 500ms in separate Tor cells (Tor's uniform-size packets). We measured the time between sending a given segment and receiving all 400 bytes.

We took measurements in the Tor network from March 13, 2009, to March 30, 2009. Every hour, we downloaded the latest router directories and used them to identify routers to test during that hour. Following the Tor-specification instruction for choosing routers, we only measured routers with the modifiers "Exit", "Fast", "Running", "Stable", and "Valid". The routers also had to be non-hibernating and could not have exit policies that disallowed connections to our host IP address and port. For every router we tested during the hour, we started an experiment after an exponentially-distributed random delay. After each experiment finished, we started another one after another exponentially-distributed random delay. There was an average number of 4.8 experiments per router per hour.

The Yale host was located at 192.31.236.5 and the server port we used was 80. The Tor client was a custom client written in Java. Packet traces were recorded using `tcpdump`. The timestamps on these were used to determine the timing of events during the measurement process.

7.2 Results

Experiments Over the course of the month, 745 routers appeared that met our selection criteria. The average number of experiments per router was 399, the median was 89, the minimum was 1, and the maximum was 1776. These numbers reflect the uptime of Tor routers. A histogram of the number of measurement experiments performed on the routers is given in Figure 2.

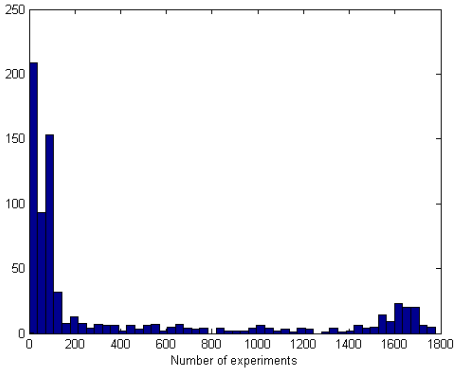


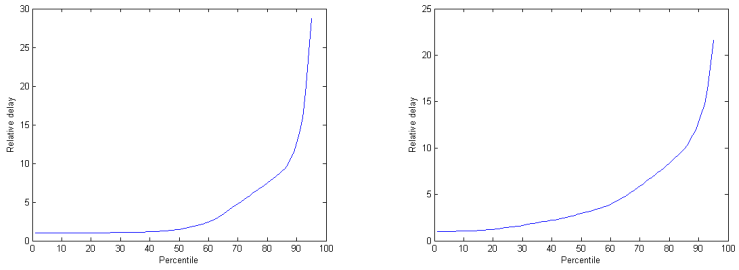
Fig. 2. Number of experiments performed on a router

Added delays From the delay measurements, we can estimate the added delay that would have resulted from our protocol. The protocol chooses a time t that must elapse from the time the message is originally sent before the message is forwarded. The time is set such that with some success probability p the packet arrives at the forwarding router in less than t time. Our data allows us to examine how much delay we would have needed to add such that a fraction p of packets would have arrived in time.

We divide the delay measurements by router and into periods of 6 hours. This period length is long enough to have 30 an expected experiments per router but is still short to adjust for the changes in average delay that occurs in many routers over time. Within each period, to achieve success probability p we set the send time t to be the smallest value that is larger than at least a fraction p of observed delays. For those delays smaller than t, using the send time t adds some delay. We look at the relative increase, *i.e.*, the total new delay divided by the original delay.

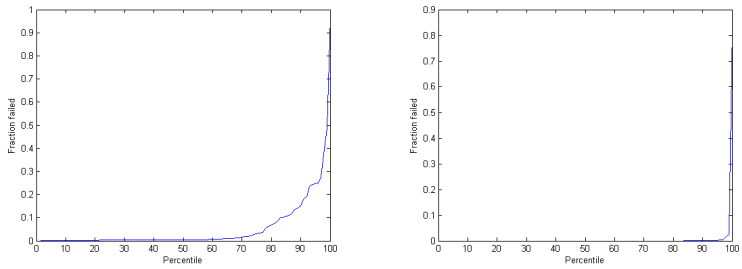
The distribution of relative connection delays, over all successfully-opened streams, to achieve a success probability of $p = 0.95$ is shown in Figure 3(a). At the 50th percentile, the relative connection delay is less than 1.48. The distribution of relative packet delays, over all successfully-delivered packets, appears in Figure 3(b). At the 50th percentile, the relative packet delay is just over 2.95.

Failures We also must consider the failure rate of opening connections and delivering packets. We could simply consider failures to be connections and packets with infinite delay, but failures can occur for reasons that have nothing to do with resource congestion. In our experiments, connection failures occur after a Tor circuit has been successfully created with the destination router. Similarly, packet-delivery failures occur after a circuit and stream have both been successfully opened. It thus seems unlikely that most of these failures are due to resource constraints. Therefore, it is useful to keep them separate.



(a) Relative connection delays, p=0.95 (b) Relative packet delays, p=0.95

Fig. 3. Relative delays



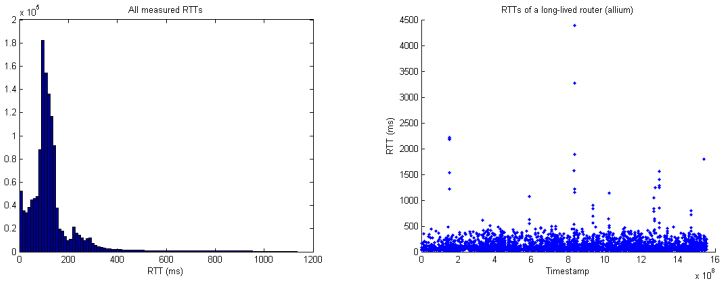
(a) Stream-open failures (b) Packet-delivery failures

Fig. 4. Measurement failures

The distribution of connection failure rates over all periods is shown in Figure 4(a). At the 50th percentile, we observe a failure rate of less than 0.005. The distribution of packet-delivery failure rates is shown in Figure 4(b). The rates stay below 0.005 until the 99th percentile. The reason for this is that packet sends are not even attempted when the connection fails. This shows that failures are much more likely when a circuit is being established than when sending over an established circuit. Circuit-establishment requires few resources, and thus failures are potentially due to factors (*e.g.* implementation errors) that do not fundamentally constrain the performance of our protocol.

Absolute delays The relative delays are useful to understand the slowdown added by our protocol. Absolute delays do matter in many applications, though, and so we present the actual delays measured on the Tor network.

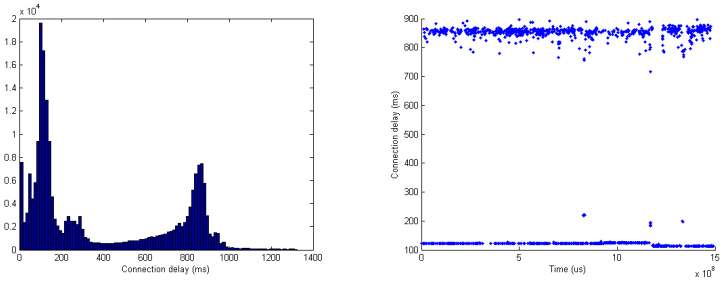
A histogram of round-trip times for all experiments is shown in Figure 5(a). The mean of these times is 161ms and the median is 110ms. Similar to the connection delays, we see a peak bin centered at 100ms. The RTTs over time for a long-lived router named *allium* appear in Figure 5(b). We can see that the distribution of RTTs in Figure 5(a) also appears among just the RTTs of a



(a) Round-trip times (w/o top 1%) (b) Round-trip times - *allium*

Fig. 5. Round-trip times

single router. These data are consistent with Mukherjee [25], where RTTs are shown to be well-modeled by a gamma distribution. In particular, these delays appear unimodal.



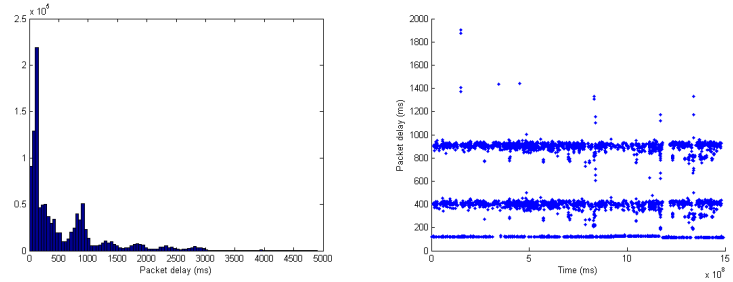
(a) Connection delays (w/o top 5%) (b) Connection delays - *b0b3r*

Fig. 6. Connection delays

A histogram of all the connection delays measured is shown in Figure 6(a). The top 5% of delays have been removed to show the rest in greater detail. It shows that nearly all delays are less than 1s. Also, we can see that the distribution is bimodal, with peaks at about 120ms and 860ms.

This pattern is present when considering the connection delays per router. We group the delays up to 1s into bins of size 20ms. Then we examine the top three most-populated bins for evidence of bimodality. We find 100 routers for which there exist, among these three bins, both one centered below 300ms and one centered above 700ms. The connection delays over time for a one such router - *b0b3r* (193.221.122.229) - is shown in Figure 6(b). The clear timing stratification suggests a cause other than varying Internet-route congestion or host-resource contention. Furthermore, the RTTs are unimodal, and so the cause is likely to

be in the Tor routers. We believe that this is due to read/write rate-limiting that Tor servers manage by periodically filling global token buckets.



(a) Packet delays (w/o top 1%) (b) Packet delays - *b0b3r*

Fig. 7. Packet-delivery delays

The delays of all 400-byte packets that were successfully forwarded is shown in Figure 7(a). We again see times that cluster around certain levels. Here, there are six obvious levels, approximately centered at 125ms, 860ms, 1350ms, 1840ms, 2330ms, and 2820ms. To examine any such clustering per router, we group the packet delays for each router by 20ms intervals from 0-5000ms and look at the largest six such groups. There are 136 routers with two groups of these six within 100ms of the observed cluster peaks. There are 14 routers with three such groups. The delay time-series of the router *b0b3r*, which exhibits clustering, is shown in Figure 7(b). As with connection delays, we see that the different levels are interleaved. This phenomenon is probably due to the same mechanism underlying the similar pattern in connection delays.

The processing delay at a router is due to both an inherent computational cost for processing packets and to resource contention. We can estimate how much of the processing cost is inherent to the Tor protocol, and by similarity, to our protocol, by deriving a distribution for processing times from our measurements.

We can combine the RTT and connection delay measurements to estimate the distribution of host processing delays. For each router, we estimate the RTT and connection-delay distributions by partitioning delay times from 0 to 1000ms into 50 bins and treating the normalized frequency counts as probabilities. The connection delay is the sum of the RTT and the processing delay, and therefore, if we assume independence of the RTT and processing-delay distributions, we can derive the processing delay distribution for each router. The distribution over all routers is shown in Figure 8. In the result, there is almost a 40% probability of having a processing delay of nearly zero. Thus processing delays are not due to the inherent cost of computation but instead to limited resources at the routers.

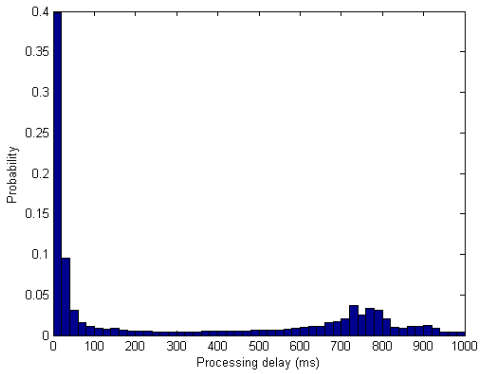


Fig. 8. Processing-delay distribution

8 Future Work

There are several developments that fit within our approach and have to potential to make it truly useful and effective.

Foremost among these is to design and evaluate a usable padding scheme. We would like it to provide large anonymity sets and low overhead. It should also allow the return padding scheme to be predetermined. It could be possible to weaken the total predictability of return traffic by allowing the user to send updates of the return padding scheme to the return path. One would have to be careful to maintain a large set of other users providing the same updates, however.

We would also like to consider choosing layered meshs and return paths freely, rather than limiting users to a fixed set of cascades. Free routes have the potential to make the anonymity set size arbitrarily large as the population grows. It seems possible to give users too much freedom, however, as each layer gives a user many potential ways to make himself observably different from the other users.

In addition, the cryptographic details of the protocol must be more carefully considered to make this protocol acceptable. We have avoided for now optimizing the efficiency of processing at the routers. Onion routing, in particular, has developed good techniques to make this aspect of the protocol fast. For example, we could use persistant circuits to speed up detecting duplicate packets. We could also switch to private keys to speed up decryption. Furthermore, a rigorous cryptographic analysis of the protocol is necessary to verify our rather informal description of its security properties.

Our analysis could be improved in some other areas as well. First, we have considered benign link delays to be failures to evaluate their effect on anonymity. However, the protocol calls for such late arrivals to be immediately forwarded. The result of such forwarding seems likely to, at some point, lead to the packet

catching up to one of its send times. Understanding this process better could improve the expected anonymity of the protocol. Also, we have observed that most of the delays measured appear to be due to congestion at the router. Tor is not optimized for latency, and therefore we should seek to understand the underlying resource issues in such networks to more accurately predict the added latencies of our protocol.

References

1. Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Information Hiding, 4th International Workshop (IH 2001)*, pages 245–257, 2001.
2. Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against Tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES 2007)*, pages 11–20, 2007.
3. Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45, 2000.
4. Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007)*, pages 92–102, 2007.
5. Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In *Advances in Cryptology Conference – CRYPTO 2005*, pages 169–187, 2005.
6. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, 1(1):65–75, 1988.
7. Wei Dai. Pipenet 1.1. Post to Cypherpunks mailing list, November 1998.
8. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 2–15, 2003.
9. Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *5th Workshop on the Economics of Information Security (WEIS 2006)*, 2006.
10. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.
11. Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *Privacy Enhancing Technologies, 4th International Workshop (PET 2004)*, pages 186–206, 2004.
12. Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. In *Proceedings of Financial Cryptography (FC '02)*, March 2002.
13. Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Preventing active timing attacks in low-latency anonymous communication. Technical Report TR-10-15, The University of Texas at Austin, 2010. ftp://ftp.cs.utexas.edu/pub/techreports/TR-1965.pdf.
14. Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Active traffic analysis attacks and countermeasures. In *2003 International Conference on Computer Networks and Mobile Computing (ICCNMC'03)*, pages 31–39, 2003.

15. Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *Proceedings of the 2003 International Conference on Parallel Processing*, pages 483–492, 2003.
16. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In *Information Hiding, First International Workshop (IH 1996)*, pages 137–150, 1996.
17. Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-TIN. How much anonymity does network latency leak? *ACM Transactions on Information and System Security*, 13(2):1–28, 2010.
18. Jan Iwanik, Marek Klonowski, and Miroslaw Kutylowski. DUO–onions and hydra–onions – failure and adversary resistant onion protocols. In *Communications and Multimedia Security: 8th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, pages 1–15, 2004.
19. Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go-MIXes providing probabilistic anonymity in an open system. In *Information Hiding, Second International Workshop (IH 1998)*, pages 83–98, 1998.
20. Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems (extended abstract). In *Financial Cryptography, 8th International Conference (FC '04)*, pages 251–265, 2004.
21. Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the Tor network. In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 63–76, 2008.
22. Jon McLachlan and Nicholas Hopper. Don't clog the queue: Circuit clogging and mitigation in P2P anonymity schemes. In *Proceedings of Financial Cryptography (FC '08)*, January 2008.
23. David Mills. Network time protocol (version 3) specification, implementation. RFC 1305, Internet Engineering Task Force, March 1992.
24. Prateek Mittal and Nikita Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 267–278, 2008.
25. Amarnath Mukherjee. On the dynamics and significance of low frequency components of internet load. *Internetworking: Research and Experience*, 5:163–205, 1992.
26. Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (SP 2005)*, pages 183–195, 2005.
27. Steven J. Murdoch and Piotr Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *Privacy Enhancing Technologies, 7th International Symposium (PET 2007)*, pages 167–183, 2007.
28. Arjun Nambiar and Matthew Wright. Salsa: a structured approach to large-scale anonymity. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006)*, pages 17–26, 2006.
29. Lasse Øverlier and Paul Syverson. Locating hidden servers. In *2006 IEEE Symposium on Security and Privacy (SP 2006)*, pages 100–114, 2006.
30. Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29, 2000.
31. Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

32. Vitaly Shmatikov and Ming-Hsui Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Computer Security ESORICS 2006, 11th European Symposium on Research in Computer Security*, pages 18–33, 2006.
33. Paul Syverson. Onion routing for resistance to traffic analysis. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX-III)*, volume 2, pages 108–110, 2003.
34. Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114, 2000.
35. TorStatus - Tor network status. http://torstatus.kgprog.com/, April 2010.
36. Xinyuan Wang, Shiping Chen, and Sushil Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *2007 IEEE Symposium on Security and Privacy (SP 2007)*, pages 116–130, 2007.
37. Mehul Motani Wei Wang and Vikram Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 323–332, 2008.
38. Wei Yu, Xinwen Fu, Steve Graham, Dong Xuan, and Wei Zhao. DSSS-based flow marking technique for invisible traceback. In *2007 IEEE Symposium on Security and Privacy (SP 2007)*, pages 18–32, Washington, DC, USA, 2007.
39. Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. On flow correlation attacks and countermeasures in mix networks. In *Privacy Enhancing Technologies, 4th International Workshop (PET 2004)*, pages 207–225, 2004.

A Anonymity of Topologies

A.1 k Entry Routers

The adversary can compromise anonymity in only two cases:

1. He controls all k entry routers and the exit router.
2. He controls an entry router, the hop after the entry routers, and the exit router.

The first case happens with probobality b^{k+1}. The second case happens with probability $(1-(1-b)^k)b^2$. The intersection of these events has probability b^{k+2}. Therefore the probability at least one occurs is

$$b^{k+1} + (1-(1-b)^k)b^2 - b^{k+2} = b^2(1-(1-b)^k + b^{k-1}(1-b)).$$

A.2 Binary-Tree Topology

The adversary can only hope to observe delays or drop that he himself created. The chance that such an attack is successful on the send tree is the chance that the packet drops propagate up the tree. The probability that this is possible can be made arbitrarily low for small-enough values of b. As in that case, p can be set so that the probability of a sent packet arriving late is arbitrarily small. There are three events at a router r that are relevant to packet loss propagating up the tree:

$\mathcal{C}_r$ r is compromised.

$\mathcal{D}_r$ Packets from all users can be stopped from coming through r.

$\mathcal{E}_r$ Packets from a known user can be stopped from coming through r.

Let $c_1(r)$ and $c_2(r)$ be the children of r. These events at r can be described in terms of in the same events at its children.

$$\mathcal{D}_r = \mathcal{C}_r \vee (\mathcal{D}_{c_1(r)} \wedge \mathcal{D}_{c_2(r)})$$
$$\mathcal{E}_r = \mathcal{C}_r \wedge (\mathcal{E}_{c_1(r)} \vee \mathcal{E}_{c_2(r)}) \vee (\mathcal{E}_{c_1(r)} \wedge \mathcal{D}_{c_2(r)}) \vee (\mathcal{D}_{c_1(r)} \wedge \mathcal{E}_{c_2(r)})$$

We use these descriptions to give recurrence relations for the probabilities of the events. Let r be at distance i from a leaf. Let D_i be the probability of $\mathcal{D}_r$.

$$D_0 = b$$
$$D_i = b + (1-b)D_{i-1}^2$$

Let E_i be the probability of $\mathcal{E}_r$.

$$E_0 = b$$
$$E_i = bE_{i-1}(2 - E_{i-1}) + (1-b)E_{i-1}(2D_{i-1} - E_{i-1}) = E_{i-1}(2b + 2(1-b)D_{i-1} - E_{i-1})$$

The probability that the adversary can block all packets going through a router is at least b, and increasing its height in the tree increases the probability up to a limit.

Theorem 6

$$\lim_{i\to\infty} D_i = \begin{cases} \frac{b}{1-b} & b \leq \frac{1}{2} \\ 1 & b \geq \frac{1}{2} \end{cases}$$

Proof. Let D^* be the fixpoint of D_i.

$$D^* = b + (1-b)(D^*)^2 = \frac{1 - |1-2b|}{2(1-b)}$$

By the definition, D_i is strictly increasing for $b > 0$. Thus D^* is achieved in the limit.

We can use this result to determine the limit of E_i.

Theorem 7

$$\lim_{i\to\infty} E_i = \begin{cases} 0 & b \leq \frac{1}{4} \\ s\,4b - 1 & \frac{1}{4} \leq b \leq \frac{1}{2} \\ 1 & b \geq \frac{1}{2} \end{cases}$$

Proof. Replace D_i with D^* in the definition of E_i, and call it $E_i^{'}$. We can compute the fixpoints for $E_i^{'}$:

$$E^{'*} = E^{'*}(2b + 2(1-b)D^* - E^{'*}) = E^{'*} = 2b - |1-2b| \vee 0$$

We will treat these as the fixpoints for E_i, although more careful arguments need to be made to show that we can treat D_i as its limit in this way.

Observe that when $b \leq 1/4$, E_i is strictly decreasing.

$$2b + 2(1-b)D_i \leq 2b + 2(1-b)D^* \leq 1$$

Therefore, in the limit, E_i reaches the higher fixpoint for $E^{'*}$, which in this case is $E^* = 0$.

Now consider the case that $1/4 \leq b \leq 1/2$. When $E_i > 2b + 2(1-b)D^* - 1$, E_i decreases with i. When $E_i < 2b + 2(1-b)D_{i-1} - 1$, E_i is increasing. Careful limiting arguments show that as $D_i \to D^*$, E_i also approaches a limit. Because E_i increases when it is small, this limit must be the fixpoint $2b - |1-2b| = 4b-1$.

Finally, let $b \geq 1/2$. E_i is multiplied each step by $2b + 2(1-b)F_i - E_i$, which, in this case, approaches $2 - E_i$. Therefore, for large i, E_i is increasing, unless E_i is very close to 1. However, even then E_i cannot decrease too much. Choosing a convergent sequence of F_i yields a sequence of E_i that approaches 1.

To link a user with his destination, the final router, r_f, must be compromised. Also, a signal must propagate either up the send tree or down the return path. The probability of at least one of these occuring is at most $b(E_h + 1 - b^l)$. Thus, by Theorem 7, we can push it arbitrarily low when $b \leq 1/4$ by increasing h and l.